Computational fluid dynamics and object oriented analysis

Zaidi, A.; Roth, Helmut; Beale, Steven

Publisher's version / Version de l'éditeur:

Computational Fluid Dynamics Journal, 15, 1, pp. 206-210, 2006

FGH62460 *CI-07502508-0*

FGH62460 CI-07502508-0

CISTI ICIST

Document Delivery Service
in partnership with the **Canadian Agriculture Library**

Service de fourniture de Documents
en collaboration avec **la Bibliothèque canadienne de l'agriculture**

THIS IS NOT AN INVOICE / CECI N'EST PAS UNE FACTURE

KATHERINE COOK
INFORMATION SPECIALIST
INST FOR CHEM PROCESS & ENVIR TECH
NATIONAL RESEARCH COUNCIL CANADA
MONTREAL ROAD BLDG M12 RM 140
OTTAWA, ON K1A 0R6
CANADA

ORDER NUMBER:	**CI-07502508-0**
Account Number:	FGH62460
Delivery Mode:	XLB
Delivery Address:	
Submitted:	2008/07/11 19:30:03
Received:	2008/07/11 19:30:03
Printed:	2008/07/14 11:06:29

Extended	**Book**	**OPENVRTOPAC**	**CANADA**

Client Number:	KCOOK FOR MCLANCY
Title:	**CFD 2005 13TH ANNUAL CONFERENCE OF THE COMPUTATIONAL FLUID DYNAMICS SOCIETY OF CANADA JULY 31-AUGUST 2, 2005, ST. JOHN'S, NEWFOUNDLAND, CANADA.**
Author:	CFD (CONFERENCE)
DB Ref. No.:	IRN20678757
Vol./Issue:	(UNKNOWN)
Date:	2005
Pages:	(UNKNOWN)
Article Title:	COMPUTATIONAL FLUID DYNAMICS AND OBJECT ORIENTED ANALYSIS
Article Author:	A ZAIDI, H ROTH, S BEALE
Report Number:	IRN20678757
Publisher:	CFD SOCIETY OF CANADA,

Estimated cost for this 7 page document: $0 document supply fee + $0 copyright = $0

COMPUTATIONAL FLUID DYNAMICS AND OBJECT ORIENTED ANALYSIS

A. Zaidi[1], *H. Roth*[1], *S. Beale*[1]

[1] National Research Council Canada, Ottawa, Canada

Email: *Azhar.Zaidi@nrc-cnrc.gc.ca*

Abstract

This study presents an identification of a set of objects in the context of an object oriented paradigm that facilitates the rapid representation and numerical solution for computational flow physics. Mathematical representations for continuum physics provide the foundation for forming such a set of objects and operator functions. A pseudo-code example is given that demonstrates the use of object classes for swift transformation of fluid flow governing equations into numerical models.

1. Introduction

Object oriented analysis and design is well-developed and widely used in software engineering. It has been shown to be beneficial in cutting software development and maintenance costs, partly through the development of reusable code. This reusable code is a direct result of a focus on the objects of the system under analysis, as these have proven to have greater stability than other features of the system, as for example algorithms. By encapsulating the objects together with the functions that operate on them, they can be used in a variety of settings.

2. Flow Physics Objects

A primary objective for an object oriented analysis of Computational Fluid Dynamics (CFD) is thus to identify the objects. The basis for forming such a set of objects is the mathematical symbolism of continuum physics itself. It can be appreciated that such mathematical-physics symbolism encapsulates the physical concepts for succinct depiction in the form of flow governing equations and their analytic manipulations. This symbolism can, therefore, be logically transformed to provide discrete computational analogues for efficient numerical manipulations.

In this context, it is found that the CFD or Flow Physics objects can be grouped into a number of categories, chiefly dealing with: computational grids; scalar, vector, and tensor fields defined on those grids; explicit differential operators, which operate on fields and generate a new one; implicit differential operators, which operate on fields and generate an augmented coefficient matrix [**A** **b**]; linear algebra structures and operators, including coefficient matrices and linear solvers.

Encapsulation of the implicit differential operators permits straight forward, rapid, and intuitive translation of a continuum mathematical model to a numerical model. To illustrate, consider the numerical solution of the unsteady advection-diffusion equation,

$$\frac{\partial}{\partial t}(\rho\phi) + \mathrm{div}(\rho\mathbf{u}\phi) - \mathrm{div}(\Gamma\,\mathbf{grad}\,\phi) = S_\phi \quad \text{on} \quad \Omega\times[t, t+\Delta t]. \tag{1}$$

In a procedural approach, one might implement an implicit finite volume solution of this equation by writing down the discretised forms of the operators and collecting terms in the unknowns to get a single equation for the unknown at a particular location in terms of the unknowns at neighbouring locations. The full set of resulting algebraic equations could then be passed to a linear system solver.

Using an object oriented paradigm, there would be a differential operator object corresponding to each term in the partial differential equation (PDE). These operators take as input one or more fields and return an augmented matrix in some sparse storage format. Overloading the arithmetic operators (by defining, for example, a "+" operation for sparse augmented matrices) permits the coefficient matrix and right hand side to be computed automatically simply by writing down the equation in terms of the operator objects. Thus, equation (1) might be implemented in a form such as,

Aaug = Op.ImplicitEuler(rho,phi) +
Op.advection(rho,u,phi) –
Op.diffusion(gamma,phi) –
Op.VolumeIntegral(S),

where, Op is an operator object and Aaug is a sparse augmented matrix obtained by adding together the sparse augmented matrices obtained from the discrete Integro-differential operators ImplicitEuler, advection, diffusion, and the VolumeIntegral that represents the sourceterm (and which affects only the right hand side of the algebraic system). The augmented matrix Aaug is then passed to a linear solver. A high level conceptual class diagram is shown in Figure 1.

3. Examples

To further the understanding of this approach and for a quick grasp of the core ideas, some examples are now presented.

3.1 Example 1 - Diffusion Operator: div(Γgradϕ)

Using a Cartesian mesh for simplicity and following [1], we derive an FV implementation of $D(\Gamma,\phi) = \mathrm{div}(\Gamma\mathbf{grad}\phi)$. Integrating D over a control volume V we find,

$$D_V = \int_V \mathrm{div}(\Gamma\mathbf{grad}\phi)dV = \int_V \frac{\partial}{\partial x}\left(\Gamma\frac{\partial\phi}{\partial x}\right)dV + \int_V \frac{\partial}{\partial y}\left(\Gamma\frac{\partial\phi}{\partial y}\right)dV + \int_V \frac{\partial}{\partial z}\left(\Gamma\frac{\partial\phi}{\partial z}\right)dV$$

Assuming the integrand in the first integral on the right is independent of y and z, and making corresponding assumptions for the second and third terms yields,

$$D_V = \Delta y\Delta z\left[\Gamma\frac{\partial\phi}{\partial x}\right]_{x_{i-1/2}}^{x_{i+1/2}} + \Delta z\Delta x\left[\Gamma\frac{\partial\phi}{\partial y}\right]_{y_{j-1/2}}^{y_{j+1/2}} + \Delta x\Delta y\left[\Gamma\frac{\partial\phi}{\partial z}\right]_{z_{k-1/2}}^{z_{k+1/2}}$$

where $\int_V dV = \Delta x\Delta y\Delta z$.

Using centred differences to discretise the derivatives leads to

$$\begin{aligned}\widetilde{D} = {} & \Delta y\Delta z\left[\Gamma_{i+1/2,j,k}\frac{\phi_{i+1,j,k}-\phi_{i,j,k}}{\Delta x} - \Gamma_{i-1/2,j,k}\frac{\phi_{i,j,k}-\phi_{i-1,j,k}}{\Delta x}\right] \\ & + \Delta z\Delta x\left[\Gamma_{i,j+1/2,k}\frac{\phi_{i,j+1,k}-\phi_{i,j,k}}{\Delta y} - \Gamma_{i,j-1/2,k}\frac{\phi_{i,j,k}-\phi_{i,j-1,k}}{\Delta y}\right] \\ & + \Delta x\Delta y\left[\Gamma_{i,j,k+1/2}\frac{\phi_{i,j,k+1}-\phi_{i,j,k}}{\Delta z} - \Gamma_{i,j,k-1/2}\frac{\phi_{i,j,k}-\phi_{i,j,k-1}}{\Delta z}\right].\end{aligned}$$

Here $\Gamma_{i+1/2,j,k}$ is some combination of $\Gamma_{i,j,k}$ and $\Gamma_{i+1,j,k}$, with corresponding combinations for the other discrete Γ values. Rearranging the terms and writing the right side as a matrix-vector multiplication, we have,

$$\widetilde{D} = \left[a_L\ a_S\ a_W\ a_P\ a_E\ a_N\ a_H\right]_{i,j,k}\begin{bmatrix}\phi_{i,j,k-1}\\ \phi_{i,j-1,k}\\ \phi_{i-1,j,k}\\ \phi_{i,j,k}\\ \phi_{i+1,j,k}\\ \phi_{i,j+1,k}\\ \phi_{i,j,k+1}\end{bmatrix},$$

$$\text{where}\begin{cases}a_L = \Gamma_{i,j,k-1/2}\Delta x\Delta y/\Delta z\\ a_S = \Gamma_{i,j-1/2,k}\Delta z\Delta x/\Delta y\\ a_W = \Gamma_{i-1/2,j,k}\Delta y\Delta z/\Delta x\\ a_E = \Gamma_{i+1/2,j,k}\Delta y\Delta z/\Delta x\\ a_N = \Gamma_{i,j+1/2,k}\Delta z\Delta x/\Delta y\\ a_H = \Gamma_{i,j,k+1/2}\Delta x\Delta y/\Delta z\\ a_P = -(a_L + a_S + a_W + a_E + a_N + a_H).\end{cases}$$

The coefficients $[a_n]_{i,j,k}$ above represent the contribution of the diffusion operator to the coefficients for $\phi_{i,j,k}$ in the sparse matrix for ϕ.

Remark 1: Nodes adjacent to boundaries have at least one boundary value for a neighbour. Coefficients for these values do not appear in the coefficient matrix **A**. Instead, these coefficients combine with the boundary values and move to the right hand side **b** of the equation; Hence, the need for the augmented matrix [**A b**].

Remark 2: In practice, it will be necessary to have choice from several different discrete possibilities for the calculation of $\Gamma_{i+1/2,j,k}$ etc.

Remark 3: Other discrete spatial differential operators in an equation for ϕ will also generate an augmented matrix. The sum of the matrices from all the operators gives the required coefficient matrix for the discrete equation.

Remark 4: Similar operators can also be defined in generalised co-ordinate systems.

3.2 Example 2: Implicit Euler Operator

Consider the equation $\frac{\partial}{\partial t}(\rho\phi) + F(\phi) = S_\phi$, where F is a function of ϕ and its spatial derivatives. Integrating over a control volume V in which ρ, ϕ are spatially invariant and approximating the time derivative by a forward difference leads to

$$\rho\frac{\phi^{(n+1)} - \phi^{(n)}}{\Delta t}\Delta x\Delta y\Delta z + \widetilde{F}\left(\phi^{(n+1)}\right) = b,$$

where, superscripts (n) and $(n+1)$ refer to neighbouring time steps, $\widetilde{F}\left(\phi^{(n+1)}\right)$ is a discrete version of $F(\phi)$ integrated over V at time $t^{(n+1)}$, and similarly b is a discrete, volume-integrated S_ϕ. Using $\Delta V = \Delta x\Delta y\Delta z$ and rearranging, we get,

$$\rho\frac{\Delta V}{\Delta t}\phi^{(n+1)} + \widetilde{F}\left(\phi^{(n+1)}\right) = b + \rho\frac{\Delta V}{\Delta t}\phi^{(n)}.$$

Thus the implicit Euler operator contributes to the diagonal coefficient (a_P) and also to the right hand side in the matrix equation $A\phi = b$. Accordingly, this operator also returns an augmented matrix Aaug[**A b**]:

Aaug = implicitEuler(ρ, ϕ) :

$$\begin{cases}\text{Aaug.A}: (a_P)_{i,j,k} \leftarrow \rho_{i,j,k}\,\Delta V/\Delta t,\\ \text{Aaug.b}: (b_P)_{i,j,k} \leftarrow \rho_{i,j,k}\,(\Delta V/\Delta t)\phi^{(n)}_{i,j,k}.\end{cases}$$

3.3. Example 3: Convection - Diffusion Usage

We use the objects to solve the equation

$$\frac{\partial}{\partial t}(\rho\phi) + \text{div}(\rho\mathbf{u}\phi) - \text{div}(\Gamma\,\mathbf{grad}\,\phi) = S_\phi \quad \text{on} \quad \Omega \times [t, t+\Delta t]$$

3.3.1. Algorithm:

- instantiate and initialise the fields ρ, ϕ, **u**, Γ, S_ϕ
- instantiate and initialise a right-hand-side vector, b

  b = VolumeIntegral (S_ϕ)
- declare an augmented matrix, Aaug
- Aaug = implicitEuler(ρ, ϕ) + advection(ρ, **u**, ϕ) − diffusion(Γ, ϕ)
- instantiate a solution vector, x ← ϕ

- GMRES(Aaug.A, x, b+Aaug.b, …)
- $\phi \leftarrow$ x

3.3.2. Pseudo-Code:

```
// Set-up a grid

Grid                mygrid(grid_dims, node_coords,
grid_origin, ….);


// Establish the fields. Initialise from pre-defined
//constants or arrays

field    rho(mygrid, rho_0);

field    phi(mygrid, phi_0);

field    U(mygrid, u0, v0, w0);

field    Gamma(mygrid, gamma_0);

field    f(mygrid, f_0);

field    F(f);


// Set Dirichlet conditions for phi on all boundary
//points

// Set the values from previously defined arrays for
//each boundary

phi.setBCType('N', 'S', 'E', 'W', 'U', 'D',
'Dirichlet');

phi.setBCValue('N', phiN);

        .

        .

phi.setBCValue('D', phiD);


// Get an explicit field operator to integrate f over V
//for F

FVExplicitFieldOperator   ExplOp;

F = ExplOp.VolumeIntegral(f);


// Get an Implicit Field Operator to get the
//coefficients and RHS updates and an

//augmented matrix to put them in

FVImplicitFieldOperator   ImplOp;

AugmentedMatrix           Aaug;

// Compute the Augmented matrix

        Aaug = ImplOp.Euler(rho, phi) +
ImplOp.Advection(rho, U, phi) +
ImplOp.Diffusion(Gamma, phi) - F;


// Set-up the Linear Algebra Solver

Solver            mysolver;

mysolver.setGMRESparams(restart, maxiterations,
tol);

converged = mysolver.GMRESsolve(Aaug.A, phi,
Aaug.b);
```

A flow schematic that visualises the interaction between object-classes is shown in Figure 2.

4. Discussion

The operators, once in place, offer obvious savings of effort and reduction of error risk in the discretisation step. Clearly, it is also easy to determine from the code what PDE is being implemented without a lot of possibly tedious reverse engineering.

What is also obvious is that a great deal of design work is necessary to ensure a rich, extendable suite of operators useable to the community, including a variety of spatial and temporal discretisation schemes (within a particular discretisation method, for example, Finite Volume, Finite Element, etc.). The required design effort and the corresponding development work required to build usable machinery before one can solve an actual problem can be a formidable impediment to the adoption of object oriented techniques. Fortunately, there are those in the community that have made it their work to carry out this development, leaving us to apply it to our own needs [2], [3], [4], and [5].

The object oriented approach also furnishes a useful pedagogical tool. In principle, scientists and students can quickly implement a numerical model of a phenomenon of interest from the mathematics. Use of differential operators that encapsulate the same mathematical operator but use different spatial or temporal discretisations can quickly show advantages and disadvantages to students of numerics, and quickly dispel the notion that these operators can be used indiscriminately by the numerically naïve. In practice, however, there is a steep learning curve to familiarity with an extensive class hierarchy.

5. Concluding Remarks

Programming in a traditional procedural paradigm, the numerical methods for CFD are usually expressed as a procedural manipulation of data primitives such as integers, floats, and arrays etc. However, the mathematical quantities used in scientific expressions such as in CFD, are much more abstract, for example, vector quantities and their associated vector fields. These fields are defined over domains, which in their corresponding software representation appear in a discretised form called a grid or mesh. In an "object-oriented" environment, such as in C++, it is possible to define and manipulate mathematical-physics abstractions through object-classes, which reflect the abstractions with a significantly higher fidelity.

A pseudo-code example was provided to demonstrate the usage of object classes to enable quick, easy and intuitive translation of fluid flow mathematical models into CFD numerical models through the encapsulation of the common mathematical operators and the data on which they act.

Further, it is strongly felt that the presented approach provides an important pedagogical paradigm with a focus on the physics of a given phenomena that has a direct correspondence with its discrete computational representation.

Finally, it should be noted that the method is extendable and can be applied to a wide range of problems and phenomena beyond CFD and continuum flow physics.

References

1. Patankar, S. V., *Numerical Heat Transfer and Fluid Flow*, 1980, New York, Hemisphere.

2. Brown, D. L., Chesshire, G. S., Henshaw, W. D., and Quinlan, D. J., *OVERTURE: An Object-Oriented Software System for Solving Partial Differential Equations in Serial and Parallel Environments*, Los Alamos Unclassified Report LA-UR-97-335, Los Alamos, New Mexico, USA.

3. http://www.llnl.gov/casc/Overture/, accessed: 10th June 2005.

4. Weller, H. G., Tabor, G., Jasak, H., and Fureby, C., *A Tensorial Approach to Computational Continuum Mechanics Using Object-Oriented Techniques*, Computers in Physics, Volume 12, No. 6, November / December 1998, American Institute of Physics.

5. http://www.openefd.co.uk/openfoam/, accessed: 10th June 2005.

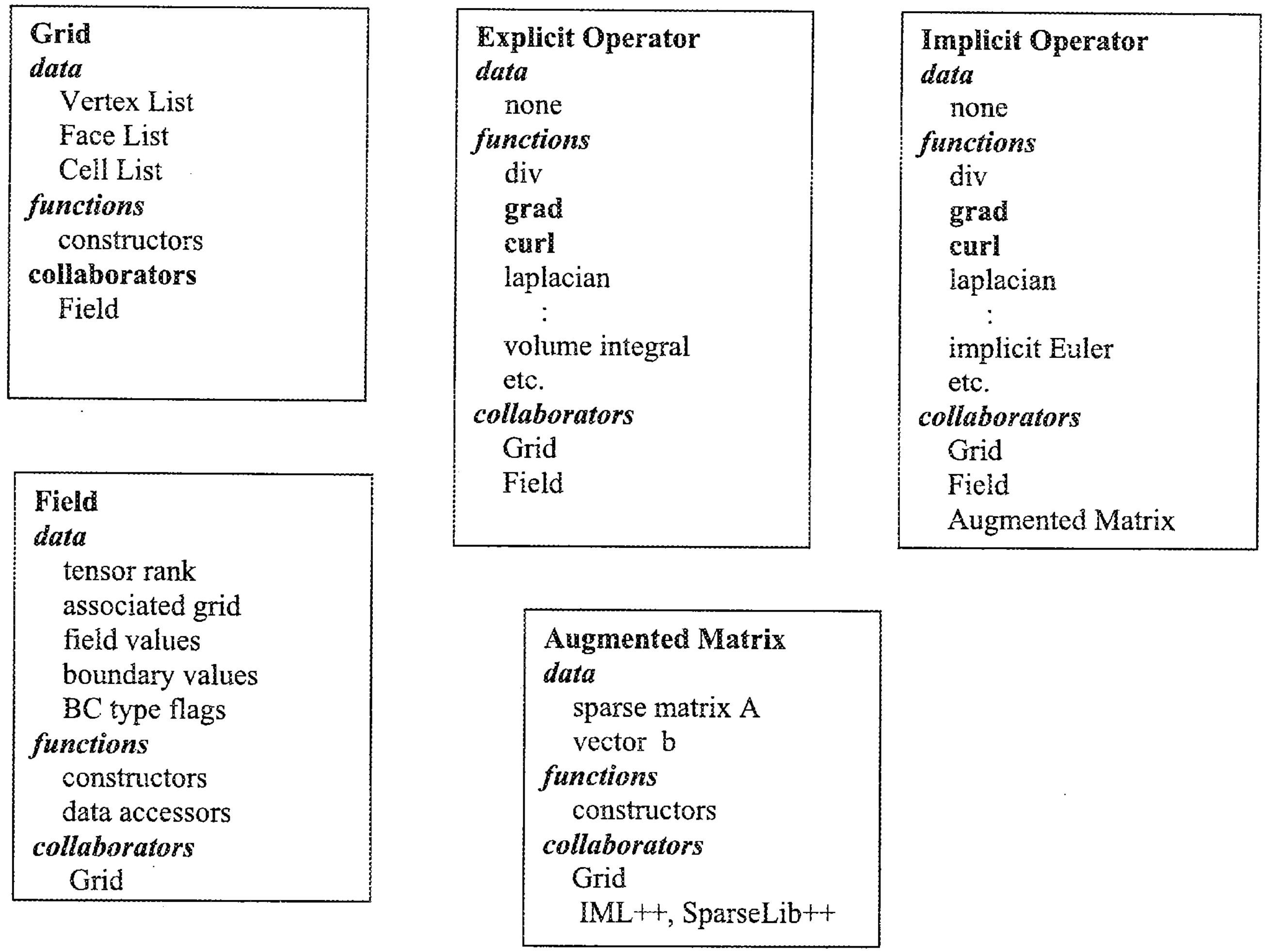

Figure 1: High Level Conceptual Class Diagram.

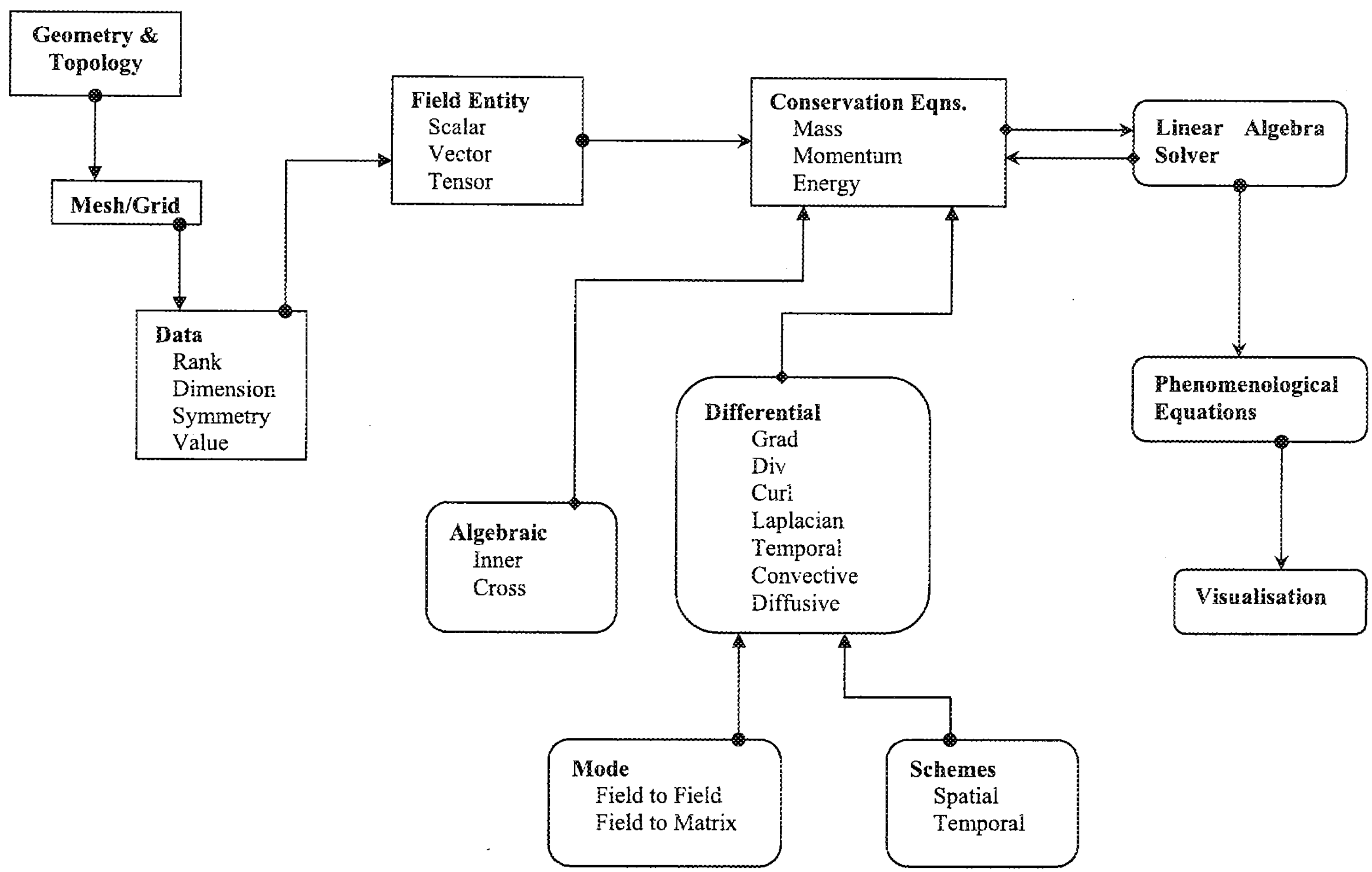

Figure 2: Object-Classes Interaction Flow Schematic.